More Randomization

Design and Analysis of Algorithms
Andrei Bulatov

Contention Resolution in a Distributed System

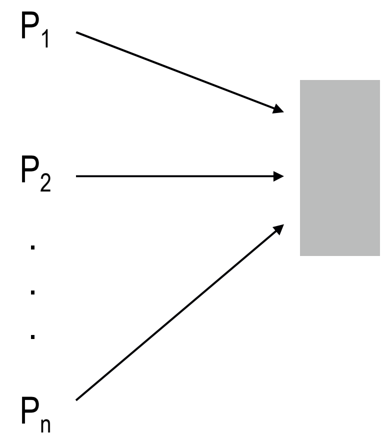
Contention resolution

Instance

Given n processes $P_1, \ldots, P_n$, each competing for access to a shared database. If two or more processes access the database simultaneously, all processes are locked out.

Objective:

Devise a protocol to ensure all processes get through on a regular basis.



Restriction: Processes cannot communicate.

Challenge. Need symmetry-breaking paradigm.

Contention Resolution: Randomized Protocol

Protocol:

Each process requests access to the database at time t with probability p = 1/n.

Claim

Let S[i, t] denote the event that process i succeeds in accessing the database at time t. Then

$$\frac{1}{e \cdot n} \le \Pr[S(i,t)] \le \frac{1}{2n}$$

Proof

process i requests access

none of remaining n-1 processes request access

By independence, $\Pr[S(i,t)] = p(1-p)^{n-1}$

Setting p = 1/n, we have $\Pr[S(i,t)] = \frac{1}{n}\underbrace{(1-\frac{1}{n})^{n-1}}_{\text{between } 1/e \text{ and } 1/2}$

value that maximizes Pr[S(i, t)]

Contention Resolution: Randomized Protocol

Useful facts from calculus.

As n increases from 2, the function:

- $\left(1-\frac{1}{n}\right)^n$ converges monotonically from 1/4 up to 1/e
- $\left(1-\frac{1}{n}\right)^{n-1}$ converges monotonically from 1/2 down to 1/e.

Claim

The probability that process i fails to access the database in e·n rounds is at most 1/e. After e·n·(c ln n) rounds, the probability is at most n^{-c}

Contention Resolution: Randomized Protocol

Proof

Let F[i, t] be the event that process i fails to access database in rounds 1 through t. By independence and previous claim, we have

$$\Pr[F(i,t)] \le \left(1-\tfrac{1}{en}\right)^{t}$$

– Choose $t = \lceil e \cdot n \rceil$: $\Pr[F(i,t)] \le \left(1-\tfrac{1}{en}\right)^{\lceil en \rceil} \le \left(1-\tfrac{1}{en}\right)^{en} \le \tfrac{1}{e}$

– Choose $t = \lceil e \cdot n \rceil \lceil c \ln n \rceil$: $\Pr[F(i,t)] \le \left(\tfrac{1}{e}\right)^{c \ln n} = n^{-c}$

Contention Resolution: Randomized Protocol

Claim

The probability that all processes succeed within 2e·n·ln n rounds is at least 1 – 1/n.

Proof

Let F[t] be the event that at least one of the n processes fails to access database in any of the rounds 1 through t.

$$\Pr[F[t]] = \Pr\left[\bigcup_{i=1}^{n} F[i,t]\right] \underbrace{\le}_{\text{union bound}} \sum_{i=1}^{n} \Pr[F[i,t]] \underbrace{\le}_{\text{previous slide}} n\left(1-\tfrac{1}{en}\right)^{t}$$

Union bound: Given events $E_1, \ldots, E_n$, $\Pr\left[\bigcup_{i=1}^{n} E_i\right] \le \sum_{i=1}^{n} \Pr[E_i]$

Contention Resolution: Randomized Protocol

Choosing $t = 2\lceil en \rceil \lceil \ln n \rceil$ yields $\Pr[F[t]] \le n \cdot n^{-2} = 1/n$

QED

Global Minimum Cut

Global min cut

Instance

A connected, undirected graph G = (V, E)

Objective

Find a cut (A, B) of minimum cardinality.

Applications.

Partitioning items in a database, identify clusters of related documents, network reliability, network design, circuit design, TSP solvers.

Global Minimum Cut

Network flow solution:

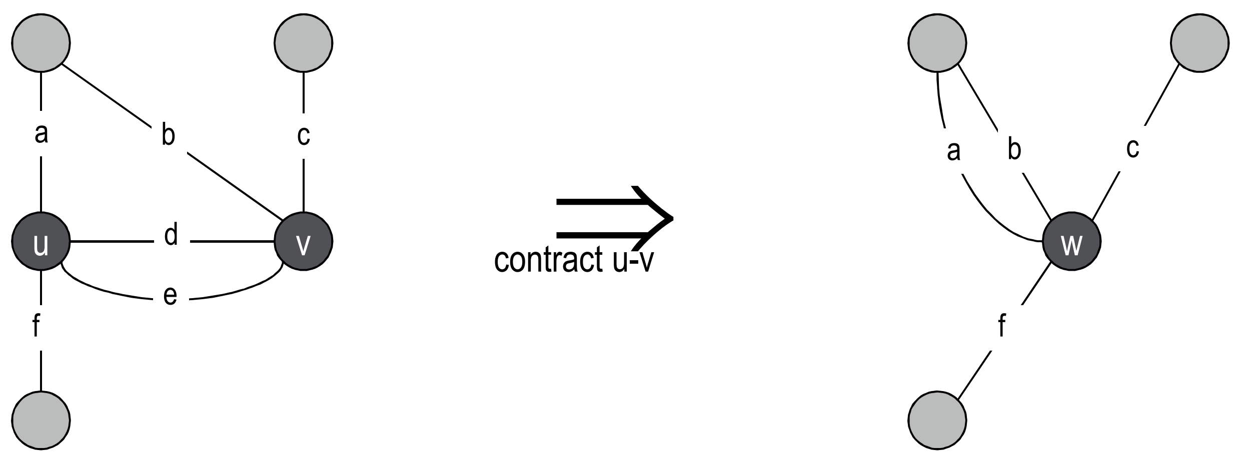
- Replace every edge (u, v) with two antiparallel edges (u, v) and (v, u).
- Pick some vertex s and compute min s-v cut separating s from each other vertex $v \in V$.

False intuition: Global min-cut is harder than min s-t cut.

Contraction Algorithm

Contraction algorithm ([Karger 1995]):

- Pick an edge e = (u,v) uniformly at random.
- Contract edge e.
  - replace u and v by single new super-node w
  - preserve edges, updating endpoints of u and v to w
  - keep parallel edges, but delete self-loops
- Repeat until graph has just two nodes v_1 and v_2
- Return the cut (all nodes that were contracted to form v_1).

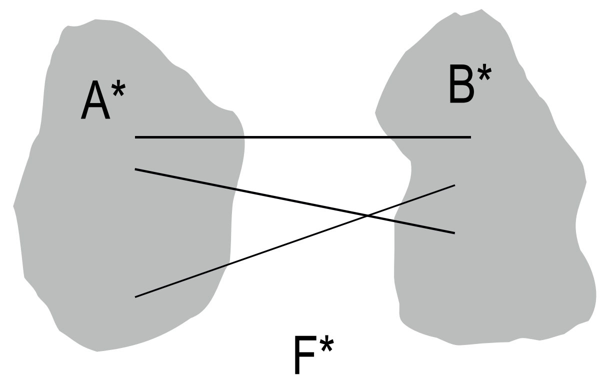


Contraction Algorithm

Claim

The contraction algorithm returns a min cut with probability $\geq \frac{2}{n^2}$

Proof



Consider a global min-cut (A*, B*) of G.

Let F* be edges with one endpoint in A* and the other in B*. Let k = |F*| = size of min cut.

In first step, algorithm contracts an edge in F* with probability k/|E|.

Every node has degree $\geq$ k since otherwise (A*, B*) would not be min-cut. $\Rightarrow$ $|E| \geq \frac{1}{2}kn$.

Thus, algorithm contracts an edge in F* with probability $\leq 2/n$.

Contraction Algorithm

Let E_j be the event that an edge in F^* is not contracted in iteration j.

$$
\begin{aligned}
\Pr[E_1 \cap E_2 \cdots \cap E_{n-2}] \\
&= \Pr[E_1] \times \Pr[E_2 \mid E_1] \times \cdots \times \Pr[E_{n-2} \mid E_1 \cap E_2 \cdots \cap E_{n-3}] \\
&\geq \left(1 - \tfrac{2}{n}\right)\left(1 - \tfrac{2}{n-1}\right) \cdots \left(1 - \tfrac{2}{4}\right)\left(1 - \tfrac{2}{3}\right) \\
&= \left(\tfrac{n-2}{n}\right)\left(\tfrac{n-3}{n-1}\right) \cdots \left(\tfrac{2}{4}\right)\left(\tfrac{1}{3}\right) \\
&= \frac{2}{n(n-1)} \\
&\geq \frac{2}{n^2}
\end{aligned}
$$

Contraction Algorithm: Amplification

Amplification:

To amplify the probability of success, run the contraction algorithm many times.

Claim.

If we repeat the contraction algorithm $n^2 \ln n$ times with independent random choices, the probability of failing to find the global min-cut is at most $1/n^2$.

Proof

By independence, the probability of failure is at most

$$\left(1-\frac{2}{n^2}\right)^{n^2 \ln n} = \left[\left(1-\frac{2}{n^2}\right)^{\frac{1}{2}n^2}\right]^{2\ln n} \leq \left(e^{-1}\right)^{2\ln n} = \frac{1}{n^2}$$

$(1 - 1/x)^x \leq 1/e$

Contraction Algorithm: The context

Remark:

Overall running time is slow since we perform $\Theta(n^2 \log n)$ iterations and each takes $\Omega(m)$ time.

Improvement: (Karger-Stein 1996) $O(n^2 \log^3 n)$.

- Early iterations are less risky than later ones: probability of contracting an edge in min cut hits 50% when $n / \sqrt{2}$ nodes remain.
- Run contraction algorithm until $n / \sqrt{2}$ nodes remain.
- Run contraction algorithm twice on resulting graph, and return best of two cuts.

Extensions: Naturally generalizes to handle positive weights.

Best known: [Karger 2000] $O(m \log^3 n)$

faster than best known max flow algorithm or deterministic global min cut algorithm

Expectation

Expectation.

Given a discrete random variables X, its expectation E[X] is defined by:

$$E[X] = \sum_{j=0}^{n} v_j \cdot \Pr[X = v_j]$$

Example

Waiting for a first success. Coin turns up heads with probability p and tails with probability 1 – p . How many independent flips X are needed until first heads?

$$E[X] = \sum_{j=0}^{\infty} j \cdot \Pr[X = j] = \sum_{j=0}^{\infty} j(1-p)^{j-1} p = \frac{p}{1-p} \sum_{j=0}^{\infty} j(1-p)^{j}$$

$$= \frac{p}{1-p} \cdot \frac{1-p}{p^2} = \frac{1}{p}$$

j-1 tails   1 head

Expectation: Two Properties

Lemma

If X is a 0/1 random variable, E[X] = Pr[X = 1].

Proof

$$E[X] = \sum_{j=0}^{n} v_j \cdot \Pr[X = v_j] = \sum_{j=0}^{1} j \cdot \Pr[X = j] = \Pr[X = 1]$$

Linearity of expectation.

Given two random variables X and Y defined over the same sample space, E[X + Y] = E[X] + E[Y]

Guessing Cards

Game: Shuffle a deck of n cards; turn them over one at a time; try to guess each card.

Memoryless guessing: No psychic abilities; cannot even remember what has been turned over already. Guess a card from full deck uniformly at random.

Claim.

The expected number of correct guesses is 1.

Proof

Let $X_i = 1$ if i^{th} prediction is correct and 0 otherwise.

Let X be the number of correct guesses, i.e. $X_1 + \ldots + X_n$.

$E[X_i] = \Pr[X_i = 1] = 1/n$.

$E[X] = E[X_1] + \ldots + E[X_n] = 1/n + \ldots + 1/n = 1$.

Guessing Cards (cntd)

Game: Shuffle a deck of n cards; turn them over one at a time; try to guess each card.

Guessing with memory: Guess a card uniformly at random from cards not yet seen.

Claim

The expected number of correct guesses is $\Theta(\log n)$.

Proof

Let $X_i = 1$ if i^{th} prediction is correct and 0 otherwise.

Let X be the number of correct guesses, i.e. $X_1 + \ldots + X_n$.

$E[X_i] = Pr[X_i = 1] = 1 / (n - i - 1)$.

$E[X] = E[X_1] + \ldots + E[X_n] = 1/n + \ldots + 1/2 + 1/1 = H(n)$.

$\ln(n+1) < H(n) < 1 + \ln n$